

BBBBBBBBBBBB		000000000		000000000		TTTTTTTTTTTT		SSSSSSSSSS
BBBBBBBBBBBB		000000000		000000000		TTTTTTTTTTTT		SSSSSSSSSS
BBBBBBBBBBBB		000000000		000000000		TTTTTTTTTTTT		SSSSSSSSSS
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000000000		000000000		TTT	SSSSSSSSSS	
BBBBBBBBBBBB		000000000		000000000		TTT	SSSSSSSSSS	
BBBBBBBBBBBB		000000000		000000000		TTT	SSSSSSSSSS	

```
CCCCCCCC 000000 NN NN FFFFFFFF 111111 GGGGGGGG UU UU RRRRRRRR EEEEEEEEE
CCCCCCCC 000000 NN NN FFFFFFFF 111111 GGGGGGGG UU UU RRRRRRRR EEEEEEEEE
CC CC 00 00 NN NN FF 11 11 GG GG UU UU RR RR EE
CC CC 00 00 NN NN FF 11 11 GG GG UU UU RR RR EE
CC CC 00 00 NNNN NN FF 11 11 GG GG UU UU RR RR EE
CC CC 00 00 NNNN NN FF 11 11 GG GG UU UU RR RR EE
CC CC 00 00 NN NN FFFFFFFF 11 11 GG GG UU UU RRRRRRRR EEEEEEEEE
CC CC 00 00 NN NN FFFFFFFF 11 11 GG GG UU UU RRRRRRRR EEEEEEEEE
CC CC 00 00 NN NN NN NN FF 11 11 GG GG UU UU RR RR EE
CC CC 00 00 NN NN NN NN FF 11 11 GG GG UU UU RR RR EE
CC CC 00 00 NN NN NN NN FF 11 11 GG GG UU UU RR RR EE
CC CC 00 00 NN NN NN NN FF 11 11 GG GG UU UU RR RR EE
CC CC 00 00 NN NN NN NN FF 11 11 GG GG UU UU RR RR EE
CC CC 00 00 NN NN NN NN FF 11 11 GG GG UU UU RR RR EE
CCCCCCCC 000000 NN NN FF 111111 GGGGGG UUUUUUUUU RRR RR EEEEEEEEE
CCCCCCCC 000000 NN NN FF 111111 GGGGGG UUUUUUUUU RRR RR EEEEEEEEE

LL 111111 SSSSSSSS
LL 111111 SSSSSSSS
LL 11 SS
LL 11 SS
LL 11 SS
LL 11 SS
LL 11 SSSSSS
LL 11 SSSSSS
LL 11 SS
LL 11 SS
LL 11 SS
LL 11 SS
LLLLLLLLLL 111111 SSSSSSSS
LLLLLLLLLL 111111 SSSSSSSS
```

CONFIGURE
Table of contents

G 16

- PROCESS TO DYNAMICALLY CONFIGURE DEVIC 15-SEP-1984 23:46:18 VAX/VMS Macro V04-00

Page 0

(1)	140	CONFIGURE - Configure devices
(1)	240	FOUND_PROC - A process has been found by the poller
(1)	308	PROCESS_MSG - Do the work of configuring the device
(1)	399	BLDNAME
(1)	447	EXIT_HANDLER


```
0000 1      .TITLE   CONFIGURE - PROCESS TO DYNAMICALLY CONFIGURE DEVICES
0000 2      .IDENT   'V04-000'
0000 3
0000 4 *****
0000 5 *
0000 6 *    COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 *    DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 *    ALL RIGHTS RESERVED.
0000 9 *
0000 10 *   THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 *   ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 *   INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 *   COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 *   OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 *   TRANSFERRED.
0000 16 *
0000 17 *   THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 *   AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 *   CORPORATION.
0000 20 *
0000 21 *   DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 *   SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 *
0000 24 * *****
0000 25 :++
0000 26
0000 27 Facility: System configuration
0000 28
0000 29 Abstract: CONFIGURE is used to dynamically configure VAX MSCP-served and HSC-
0000 30          served disks and tapes.
0000 31
0000 32 Environment: It is run as a process, in user, exec and kernel modes.
0000 33
0000 34 Author: Maryann Hinden, Creation date: 02-JUN-1983
0000 35
0000 36 Modification History:
0000 37
0000 38         V03-004 WHM0001           Bill Matthews           11-Apr-1984
0000 39         Purge working set before hibernating.
0000 40
0000 41         V03-003
0000 42         Change value in BOO$GL_CONADP to indicate noadapter.
0000 43
0000 44         V03-002 WMC0001           Wayne Cardoza           11-Aug-1983
0000 45         Polling must be reenabled in kernel mode.
0000 46
0000 47         V03-001 MSH0001           Maryann Hinden           14-Jul-1983
0000 48         Add jacket routine BOO$CONFIGMN to image, and
0000 49         remove some code.
0000 50
0000 51 :--
0000 52
0000 53 Include files:
0000 54
0000 55 $ACFDEF
0000 56
0000 57 ; Define autoconfiguration block
```

```
0000 58 $IODEF
0000 59 $IPLDEF
0000 60 $LCKDEF
0000 61 $PRCPOLDEF ; Define process poller mailbox offsets
0000 62 $SBDEF
0000 63 $SSDEF ; System status definitions
0000 64 $SYSMSGDEF ; Sysgen messages
0000 65
0000 66
0000 67 ; Equated Symbols
0000 68
00000123 0000 69 WRTATNFLAG = <IOS_SETMODE!IOSM_WRTATTN>
00000071 0000 70 READFLAG = <IOS_READVBLK!IOSM_NOW>
0000 71
00000000 0000 72 SERVER = 0 ; Offsets into process info block
00000010 0000 73 DEVICE = 16
00000012 0000 74 DRIVER = 18
00000018 0000 75 SPPB = 27
0000 76
0000 77 ;
0000 78 ; Macros
0000 79
0000 80 .MACRO PRCINFO SERVER,DEVICE,DRIVER ; Builds process info table
0000 81
0000 82 .PSECT INFO_BLOCK ; Actual data area
0000 83
0000 84 $SERVERNAMES = .
0000 85 .ASCII / ; Reserve room for all 16 chars
0000 86 $NAMENDS = . ; Remember end of block
0000 87 . = $SERVERNAMES ; Get back to beginning
0000 88 .ASCII /SERVER/ ; Now store the real name
0000 89 . = $NAMENDS ; Go to end of block
0000 90 .ASCII /DEVICE/ ; Device name
0000 91 .ASCII /DRIVER/ ; Driver name
0000 92 .LONG 0 ; SPPB for polling this process
0000 93
0000 94 .PSECT INFO_PTR ; A list of pointers to the data
0000 95
0000 96 .LONG $SERVERNAMES
0000 97
0000 98 .ENDM
0000 99
0000 100 ; Own Storage
0000 101 ;
00000000 0000 102 .PSECT INFO_PTR
0000 103
0000 104 PROC_INFO:
0000 105 PRCINFO MSCPS$DISK,DU,DUDRIVER ; Build the process info table
0000 106 PRCINFO MSCP$TAPE,MU,TUDRIVER
0000 107
00000008 0000 108 .PSECT INFO_PTR
00000000 0000 109 .LONG 0 ; Indicate end of table
0000001C 000C 110 .BLKL 4 ; Patch area for four more processes
0000 111
0000003E 001C 112 .PSECT INFO_BLOCK
000000BA 003E 113 .BLKB <SPPB+4>*4 ; Patch area for data
0000 114 00BA
```

00000000	115	.PSECT	NONPAGED DATA,NOEXE,WRT	
00000016 0000	116	FULL_NAME:	.BLKB 22	; Storage area for cluster dev name
0000001B 0016	117	DEVNAME:	.BLKB 5	; Storage area for short dev name
001B	118			
00000000 001B	119	EXIT_BLOCK:	.LONG 0	; Data block for exit handler
00000251' 001F	120		.LONG EXIT_HANDLER	
00000001 0023	121		.LONG 1	
0000002B' 0027	122		.LONG EXIT_STATUS	
0000002F 002B	123	EXIT_STATUS:	.BLKL 1	
002F	124			
00000033' 002F	125	KARGLST:	.LONG SPPBARG	; Argument list for CANCEL_POLL
00000000 0033	126	SPPBARG:	.LONG 0	; kernel mode routine
0037	127			
00000038 0037	128	MSGBUFSIZ:	.LONG PRCPOL\$C_SIZ	; Buffer used by mailbox read
00000073 003B	129	MSGBUF:	.BLKB PRCPOL\$C_SIZ	
0073	130			
0000 0073	131	MBXCHAN:	.WORD 0	; Mailbox I/O channel
0000007D 0075	132	STATUS_BLOCK:	.BLKL 2	; I/O completion status block
007D	133			
00000000	134	.PSECT	PAGED_CODE,EXE,WRT	
0000	135	PURGE_LIMITS:		; Limits for purge working set
00000000 0000	136		.LONG 0	; Purge all of P0 and P1
7FFFFFFF 0004	137		.LONG ^X7FFFFFFF	
0008	138			


```

0008 140 .SBTTL CONFIGURE - Configure devices
0008 141 :++
0008 142 :
0008 143 PURPOSE
0008 144 To start polling on cluster members in order to find out about
0008 145 HSC- and MSCP-served devices on other systems.
0008 146 :
0008 147 INPUT
0008 148 None
0008 149 :
0008 150 OUTPUT
0008 151 None
0008 152 :
0008 153 FUNCTIONAL DESCRIPTION
0008 154 This routine requests polling on all systems in the cluster
0008 155 for all processes described in the process information table.
0008 156 The process poller communicates with the CONFIGURE process via
0008 157 a mailbox. Once the polling requests have been sent out, a
0008 158 write attention AST to the mailbox is issued, and the routine
0008 159 hibernates waiting for input.
0008 160 :
0008 161 In order to cancel polling (and clean up properly) if the image
0008 162 should terminate abnormally, this routine declares an exit handler.
0008 163 :--
001C 0008 164
0008 165 .ENTRY BOO$CONFIGURE, ^M<R2,R3,R4>
000A 166 :
000A 167 :
000A 168 Create mailbox used to communicate with process poller
000A 169 :
000A 170 $CREMBX_S prmflg = #1,-
000A 171 chan = MBXCHAN,-
000A 172 promsk = #^XFF00
60 50 E9 0025 173 BLBC R0,10$
0028 174 :
0028 175 :
0028 176 Declare exit handler to be used when image exits
0028 177 :
0028 178 $DCLEXH_S desblk = EXIT_BLOCK
50 50 E9 0035 179 BLBC R0,10$
0038 180 :
0038 181 :
0038 182 Now request polling on all processes
0038 183 :
0038 184 $CMKRNLS REQ_POLL,(AP)
3E 50 E9 0047 185 BLBC R0,10$
004A 186 :
004A 187 :
004A 188 We are finished requesting polling. Now set a write attention AST
004A 189 and hibernate while waiting for responses from the poller.
004A 190 (We assume that at least one call to SCSS$POLL_MBX was successful).
004A 191 :
004A 192 $QIO_S chan = MBXCHAN,-
004A 193 func = #WRTATNFLAG,-
004A 194 p1 = FOUND_PROC,-
004A 195 p2 = PROC_INFO
12 50 E9 0073 196 BLBC R0,10$

```

```

0076 197
0076 198      $PURGWS_S inadr = PURGE_LIMITS ; minimize system resources
0080 199      $HIBER_S
04 0087 200      RET
0088 201
0088 202
0088 203 :
0088 204 : An error occurred on the create mailbox, when calling the process
0088 205 : poller, or when issuing the QIO. Send out the error message and terminate.
0088 206 : The exit handler (if declared at this point) will clean up.
0088 207
50 007C8132 8F D0 0088 208 10$: MOVL #SYSG$ CONFIGERR,R0
    FF6E' 30 008F 209      BSBW PUTERROR
04 0092 210      $EXIT_S
009B 211      RET
009C 212 :
009C 213 : Request polling on all processes we want to know about
009C 214 :
0000 009C 215 REQ_POLL: .WORD 0
009E 216
009E 217      SETIPL #IPL$ ASTDEL
53 00000073'EF 3C 00A1 218      MOVZWL MBXCHAN,R3 ; Get channel address
54 00000000'EF 9E 00A8 219      MOVAB PROC_INFO,R4 ; Get top of process table
    52 84 D0 00AF 220      MOVL (R4)+,R2 ; Get address of first process name
    50 53 D0 00B2 221
    00000000'EF 16 00B2 222 10$: MOVL R3,R0 ; Channel # in R0 is arg to call
    00B5 223      JSB SC$SPOLL_MBX ; Request polling for this process
    00BB 224
    00BB 225 :
    00BB 226 : R1 contains address of SPPB - need later to cancel polling
    00BB 227 : R2 is preserved and points to process info block
    00BB 228 :
    0C 50 E9 00BB 229      BLBC R0,20$
    1B A2 51 D0 00BE 230      MOVL R1,SPPB(R2) ; Save SPPB
    52 84 D0 00C2 231      MOVL (R4)+,R2 ; Get next process name
    EB 12 00C5 232      BNEQ 10$ ; If NEQ, poll for it
    50 01 9A 00C7 233      MOVZBL #SS$ _NORMAL,R0 ; Indicate success
    00CA 234
    00CA 235 20$:
    00CA 236      SETIPL #0 ; Lower IPL
    04 00CD 237      RET ; Return error to caller
    00CE 238

```



```
00CE 240 .SBTTL FOUND_PROC - A process has been found by the poller
00CE 241 :++
00CE 242 :
00CE 243 PURPOSE
00CE 244 Routine which is called when the process poller mailbox has been
00CE 245 written into.
00CE 246 :
00CE 247 INPUT
00CE 248 Mailbox messages - implicit
00CE 249 :
00CE 250 OUTPUT
00CE 251 Processed messages
00CE 252 :
00CE 253 FUNCTIONAL DESCRIPTION
00CE 254 This routine is called at AST level. It first re-enables the
00CE 255 write attention AST for the mailbox. It then reads and processes
00CE 256 messages until there are none left.
00CE 257 :
00CE 258 :--
007C 00CE 259
00D0 260 .ENTRY FOUND_PROC, *M<R2,R3,R4,R5,R6>
00D0 261 :
00D0 262 :
00D0 263 Before doing anything else, we requeue the write attention AST request
00D0 264 :
00D0 265 $QIO_S chan = MBXCHAN,-
00D0 266 func = #WRTATNFLAG,-
00D0 267 p1 = FOUND_PROC,-
00D0 268 p2 = PROC_INFO
47 50 E9 00F6 269 BLBC R0,30$
00F9 270 :
00F9 271 :
00F9 272 Now, read mailbox messages until there are none left
00F9 273 :
00F9 274 10$: $QIO_S chan = MBXCHAN,-
00F9 275 func = #READFLAG,-
00F9 276 iosb = STATUS_BLOCK,-
00F9 277 p1 = MSGBUF,-
00F9 278 p2 = MSGBUFSIZ
2B 50 E9 0126 279 BLBC R0,40$
54 00000075'EF 9E 0129 280 MOVAB STATUS_BLOCK,R4 : Get address of status block
0870 8F 64 B1 0130 281 CMPW (R4),#SS$_ENDOFFILE : Have we read all the msgs?
08 13 0135 282 BEQL 20$ : If EQL, yes
1A 64 E9 0137 283 BLBC (R4),40$ : If LBC, then some sort of error
0022 30 013A 284 BSBW PROCESS_MSG : Else the poller found something
BA 11 013D 285 BRB 10$ : Look for more messages
04 013F 286 :
0140 287 20$: RET
0140 288 :
0140 289 :
0140 290 An error has occurred when trying to requeue the write attention AST.
0140 291 Have the image exit.
0140 292 :
0140 293 30$:
50 007C8132 8F D0 0140 294 MOVL #SYSG$ CONFIGERR,R0
FEB6' 30 0147 295 BSBW PUTERROR
014A 296 $EXIT_S
```

CONFIGURE
V04-000

B 1

- PROCESS TO DYNAMICALLY CONFIGURE DEVIC 15-SEP-1984 23:46:18 VAX/VMS Macro V04-00
FOUND_PROC - A process has been found by 4-SEP-1984 23:03:46 [BOOTS.SRC]CONFIGURE.MAR;1

Page 7
(1)

```
04 0153 297 RET
    0154 298
    0154 299
    0154 300 : An error has occurred when reading the mailbox message. Send out the
    0154 301 : error message and dismiss the AST.
    0154 302 :
    0154 303 40$:
50 007C8132 8F D0 0154 304 MOVL #SYSG$ CONFIGERR,R0
    FEA2' 30 015B 305 BSBW PUTERROR
    04 015E 306 RET
```

```

015F 308 .SBTTL PROCESS_MSG - Do the work of configuring the device
015F 309 ++
015F 310
015F 311 PURPOSE
015F 312 Workhorse routine to actually configure the device database
015F 313 for the server which has been found.
015F 314
015F 315 INPUT
015F 316 MSGBUF - contains the actual message
015F 317
015F 318 OUTPUT
015F 319 Configured device and driver
015F 320
015F 321 FUNCTIONAL DESCRIPTION
015F 322 This routine uses the node name (contained in the message) together
015F 323 with the information associated with the server process name to
015F 324 construct a cluster device name. It then calls the connect code
015F 325 to actually construct the device database and load the class driver.
015F 326 --
015F 327
015F 328 PROCESS_MSG:
015F 329 PUSHF #M<R2,R3,R4,R5,R6> ; Save registers touched here
016A 330 CALLS #0,BOOS$CONRESET ; Reset connect information
016A 331
016A 332 :
016A 333 Search through the list of processes we are looking for to see
016A 334 if there is a match
016A 335 :
56 0000003B'EF 9E 016A 336 MOVAB MSGBUF,R6 ; Get address of message buffer
54 00000000'EF 9E 0171 337 MOVAB PROC_INFO,R4 ; Get address of process information
0178 338
0178 339 10$: MOVL (R4)+,R5 ; Get next entry
017B 340 BEQL 20$ ; If EQL, no more entries & no match
18 A6 65 10 29 017D 341 CMPC3 #16,SERVER(R5),PRCPOL$B_PRCNAM(R6) ; Compare
0182 342 BNEQ 10$ ; If NEQ, try next one
0184 343
0184 344 :
0184 345 A match was found - save info needed for the connect call and build the
0184 346 device name
0184 347 :
00000000'EF 01 CE 0184 348 MNEGL #1,BOOS$GL CONADP ; Don't use an adapter
00000000'EF D4 018B 349 CLRL BOOS$GL_CONCUNIT ; Unit number always 0
00000000'EF D4 0191 350 CLRL BOOS$GL_CONAUNIT ; Same for adapter unit
00000000'EF 66 7D 0197 351 MOVQ PRCPOL$ SYSIDL(R6),BOOS$GL_CONSYSID ; Save the sys ID from msg
00000000'EF 12 A5 9E 0197 352 MOVAB DRIVER(R5),BOOS$GL_CONDRV ; Save the driver name from proc_info
52 08 A6 9E 01A0 353 MOVAB PRCPOL$T_MODNAM(R5),R2 ; Get node name arg from msg
55 10 A5 9E 01AA 354 PUSHF #M<R5> ; Save pointer to proc_info
55 10 10 9E 01AC 355 MOVAB DEVICE(R5),R5 ; Get device name arg from proc_info
55 10 10 01B0 356 BSBB BLDNAME ; Construct the cluster device name
20 BA 01B2 357 POPR #M<R5> ; Restore
0184 358
0184 359 :
0184 360 Connect the device - build the class device database, load the class driver,
0184 361 and initialize the device
0184 362 :
00000000'EF 00 FB 0184 363 CALLS #0,BOOS$CONNECT
12 50 E8 01BB 364 BLBS R0,15$

```



```

0C 50   E9 01BE 365      $CMKRNLS ROUTIN = 30$      ; Polling must be turned on from K mode
          01CD 366      BLBC -R0,25$
          01D0 367      :
          01D0 368      :
          01D0 369      : All done
          01D0 370      :
007C 8F   BA 01D0 371 15$: POPR      #^M<R2,R3,R4,R5,R6>      ; Restore registers touched here
          05 01D4 372      RSB
          01D5 373      :
          01D5 374      :
          01D5 375      : There was no process name match - we got a spurious mailbox message
          01D5 376      :
          01D5 377 20$:
50 007C8132 8F D0 01D5 378      MOVL      #SYSG$ CONFIGERR,R0
          FE21 30 01DC 379 25$: BSBW      PUTERRR
          007C 8F BA 01DF 380      POPR      #^M<R2,R3,R4,R5,R6>      ; Restore registers touched here
          05 01E3 381      RSB
          01E4 382      :
          01E4 383      :
          01E4 384      : There was an error connecting the device - CONNECT already let the
          01E4 385      : world know.
          01E4 386      :
          01E4 387 30$:
          0000 01E4 388      .WORD      0
          52 66 9E 01E6 389      MOVAB     PRCPOL$ SYSIDL(R6),R2      ; Get system ID
          51 1B A5 D0 01E9 390      MOVL      SPPB(R5),R1      ; Get SPPB
          50 01 9A 01ED 391      MOVZBL     #1,R0      ; Re-enable polling
          00000000 GF 16 01F0 392      SETIPL     #IPL$ SCS      ; Raise IPL
          01F3 393      JSB      G^SCS$ POLL MODE      ; Request polling again
          01F9 394      SETIPL     #IPL$ ASTDEL      ; Restore IPL
          07 50 E8 01FC 395      BLBS      R0,35$
50 007C813A 8F D0 01FF 396      MOVL      #SYSG$ CANTPOLL,R0      ; Indicate unable to restart poll on
          04 0206 397 35$: RET

```

```

0207 399 .SBTTL BLDNAME
0207 400 :++
0207 401 :
0207 402 PURPOSE
0207 403 Construct cluster device name given the node name and the
0207 404 device prefix.
0207 405
0207 406 INPUT
0207 407 R2 - Address of the node name string (in counted ASCII)
0207 408 R5 - Address of the device prefix
0207 409
0207 410 OUTPUT
0207 411 FULL_NAME_PTR - contains address of complete device name string
0207 412 BOO$GL_CONDEV - contains pointer into complete device name string,
0207 413 starting at device prefix
0207 414 All registers preserved.
0207 415
0207 416 FUNCTIONAL DESCRIPTION
0207 417 This routine builds a cluster device name of the form:
0207 418
0207 419 byte 0: count of chars in string
0207 420 1 to m: node name
0207 421 m+1: '$'
0207 422 m+2 to m+4: 'xxA', where xx is the device name used by a given server
0207 423 :--
0207 424
0207 425 BLDNAME:
0207 426 PUSH R0,R1,R2,R3,R4,R5
0209 427 MOVAB FULL_NAME,R3 ; Pointer to output buffer
0210 428 MOVAB (R3)+,G^FULL_NAME_PTR ; Set up ptr for connect
0217 429 ASSUME SB$T_NODENAME+16,EQ,SB$L_DDB ; Make sure size doesn't change
0217 430 MOVZBL (R2)+,R4 ; Get real length of string
021A 431 PUSH R2,R4,R5 ; Save regs destroyed by MOVZBL
021C 432 MOV C3,R4,(R2),(R3) ; Store node name in buffer
0220 433 POP R2,R4,R5 ; Restore regs (R3 now points to next
0222 434 ; byte in dest. buffer after node name)
0222 435 MOV B #A/$/,(R3)+ ; Set in separator
0225 436 MOV W (R5),(R3) ; Store device prefix
0228 437 MOV B #A/A/,2(R3) ; Store controller letter
022D 438 ADD B3 #4,R4,FULL_NAME ; String is ASCII
0235 439 MOVL (R3),DEVNAME+1 ; Store device name
023C 440 MOV B #3,DEVNAME ; Store count
0243 441 MOVAB DEVNAME,BOO$GL_CONDEV ; Store address of device string
024E 442 POP R0,R1,R2,R3,R4,R5
0250 443
0250 444 RSB
0251 445

```

```

0251 447 .SBTTL EXIT_HANDLER
0251 448 :++
0251 449 :
0251 450 PURPOSE
0251 451 Cancel polling on mailbox (if any) at image exit.
0251 452 :
0251 453 INPUT
0251 454 Saved SPPB addresses in PROC_INFO table.
0251 455 :
0251 456 OUTPUT
0251 457 Cancelled polling.
0251 458 :
0251 459 :--
0251 460 :
001C 0251 461 .ENTRY EXIT_HANDLER, ^M<R2,R3,R4>
0253 462 :
53 00000000'EF 9E 0253 463 MOVAB PROC_INFO,R3 ; Get address of process info table
025A 464 :
54 83 D0 025A 465 10$: MOVL (R3)+,R4 ; Point to next info block
22 13 025D 466 BEQL 20$ ; If EQL, end of table
00000033'EF 1B A4 D0 025F 467 MOVL SPPB(R4),SPPBARG ; Get address of SPPB
F1 13 0267 468 BEQL 10$ ; If EQL, we haven't polled for this process
0269 469 $CMKRNLS routin=CANCEL_POLL,- ; Cancel polling
0269 470 arglst=KARGLST
1B A4 D4 027C 471 CLRL SPPB(R4) ; Show no more polling for this process
D9 11 027F 472 BRB 10$ ; Loop through table
0281 473 :
0281 474 20$: $DELMBX_S chan=MBXCHAN ; Mark mailbox for deletion
028F 475 $CMEXEC_S routin=DQLOCKS ; Dequeue locks
50 01 3C 029E 476 MOVZWL #SS$_NORMAL,R0
04 02A1 477 RET
02A2 478 :
02A2 479 :
02A2 480 : Kernel mode routine running at IPL$_ASTDEL which cancels the polling mailbox.
02A2 481 :
0004 02A2 482 .ENTRY CANCEL_POLL,^M<R2>
02A4 483 :
02A4 484 SETIPL #IPL$_ASTDEL
51 04 AC D0 02A7 485 MOVL 4(AP),R1 ; Get SPPB address
00000000'EF 16 02AB 486 JSB SCSS$CANCEL_MBX ; Cancel polling
04 02B1 487 SETIPL #0
02B4 488 RET
02B5 489 :
02B5 490 :
02B5 491 : Exec mode routine to dequeue all locks held
02B5 492 :
0000 02B5 493 .ENTRY DQLOCKS,^M<>
02B7 494 :
02B7 495 $DEQ_S lkid = #0,-
02B7 496 flags = #LCK$_DEQALL
04 02C6 497 RET
02C7 498 :
02C7 499 .END

```


CONFIGURE
Symbol table

- PROCESS TO DYNAMICALLY CONFIGURE DEVIC 15-SEP-1984 23:46:18 VAX/VMS Macro V04-00
4-SEP-1984 23:03:46 [BOOTS.SRC]CONFIGURE.MAR;1

Page 12
(1)

\$ST1	=	00000000		
\$NAMEND\$	=	0000002F	R	03
\$SERVERNAME\$	=	0000001F	R	03
BLDNAME	=	00000207	R	05
BOOS\$CONFIGURE	=	00000008	RG	05
BOOS\$CONNECT		*****	X	05
BOOS\$CONRESET		*****	X	05
BOOS\$GL_CONADP		*****	X	05
BOOS\$GL_CONAUNIT		*****	X	05
BOOS\$GL_CONCUNIT		*****	X	05
BOOS\$GL_CONDEV		*****	X	05
BOOS\$GL_CONDRV		*****	X	05
BOOS\$GQ_CONSYSID		*****	X	05
CANCEL_POLL		000002A2	RG	05
DEVICE	=	00000010		
DEVNAME		00000016	R	04
DQLOCKS		000002B5	RG	05
DRIVER	=	00000012		
EXIT_BLOCK		0000001B	R	04
EXIT_HANDLER		00000251	RG	05
EXIT_STATUS		0000002B	R	04
FOUND_PROC		000000CE	RG	05
FULL_NAME		00000000	R	04
FULL_NAME_PTR		*****	X	05
IOSM_NOW	=	00000040		
IOSM_WRTATTN	=	00000100		
IOS_READVBLK	=	00000031		
IOS_SETMODE	=	00000023		
IPL\$ASTDEL	=	00000002		
IPL\$SCS	=	00000008		
KARG\$ST	=	0000002F	R	04
LCKSM_DEQALL	=	00000001		
MBXCHAN		00000073	R	04
MSGBUF		0000003B	R	04
MSGBUFSIZ		00000037	R	04
PR\$ IPL		*****	X	05
PRCPOL\$B_PRCNAM	=	0000001B		
PRCPOL\$C_SIZ	=	00000038		
PRCPOL\$L_SYSIDL	=	00000000		
PRCPOL\$T_NODNAM	=	00000008		
PROCESS_MSG		0000015F	R	05
PROC_INFO		00000000	R	02
PURGE_LIMITS		00000000	R	05
PUTERROR		*****	X	05
READFLG	=	00000071		
REQ_POLL		0000009C	R	05
SB\$C_DDB	=	00000054		
SB\$T_NODENAME	=	00000044		
SCSS\$CANCEL_MBX		*****	X	05
SCSS\$POLL_MBX		*****	X	05
SCSS\$POLL_MODE		*****	X	05
SERVER	=	00000000		
SPPB	=	0000001B		
SPPBARG		00000033	R	04
SS\$_ENDOFFILE	=	00000870		
SS\$_NORMAL	=	00000001		
STATUS_BLOCK		00000075	R	04

SYSS\$CMEXEC	*****	GX	05
SYSS\$CMKRN	*****	GX	05
SYSS\$CREMBX	*****	GX	05
SYSS\$DCLEXH	*****	GX	05
SYSS\$DELMXB	*****	GX	05
SYSS\$DEQ	*****	GX	05
SYSS\$EXIT	*****	GX	05
SYSS\$HIBER	*****	GX	05
SYSS\$PURGWS	*****	GX	05
SYSS\$QIO	*****	GX	05
SYSG\$_CANTPOLL	= 007C813A		
SYSG\$_CONFIGERR	= 007C8132		
WRTATN\$FLG	= 00000123		

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
INFO_PTR	0000001C (28.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
INFO_BLOCK	000000BA (186.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
NONPAGED_DATA	0000007D (125.)	04 (4.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
PAGED_CODE	000002C7 (711.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.05	00:00:00.54
Command processing	115	00:00:00.69	00:00:01.93
Pass 1	320	00:00:10.32	00:00:21.76
Symbol table sort	0	00:00:01.50	00:00:03.69
Pass 2	98	00:00:02.12	00:00:03.49
Symbol table output	9	00:00:00.12	00:00:02.67
Psect synopsis output	3	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	576	00:00:14.83	00:00:34.11

The working set limit was 1500 pages.
57546 bytes (113 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1005 non-local and 15 local symbols.
499 source lines were read in Pass 1, producing 35 object records in Pass 2.
33 pages of virtual memory were used to define 31 macros.

! Macro library statistics !

Macro library name	Macros defined
\$255\$DUA28:[BOOTS.OBJ]BOOTS.MLB;1	0
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	6
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	21
TOTALS (all libraries)	27

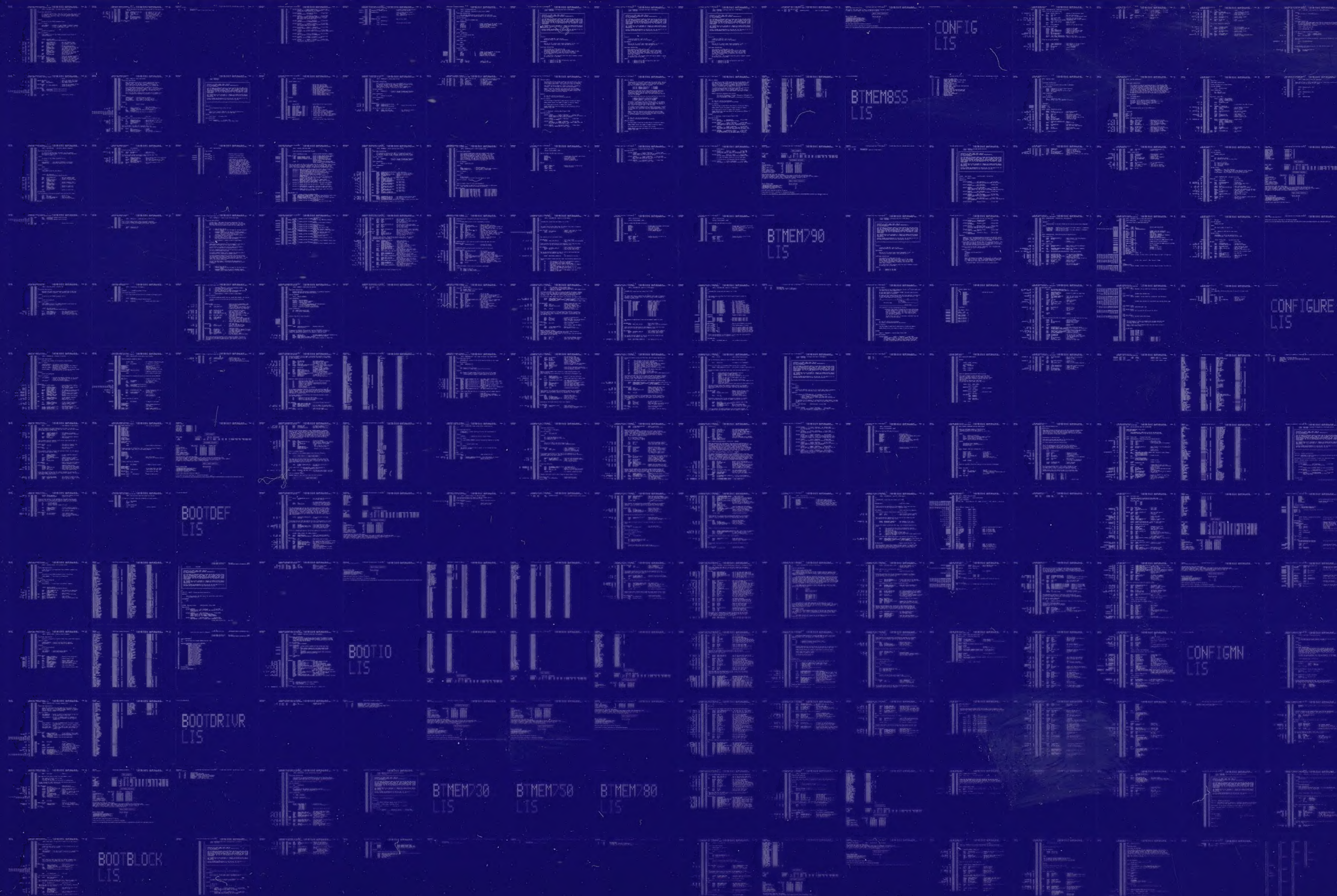
1176 GETS were required to define 27 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:CONFIGURE/OBJ=OBJ\$:CONFIGURE MSRC\$:CONFIGURE/UPDATE=(ENH\$:CONFIGURE)+EXECMLS/LIB+LIB\$:BOOTS.MLB/LIB

0037 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0038 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY